



AL1x2x Add-On Instruction Guide for Acyclic Commands Allen Bradley Compact/ControlLogix PLC's

ThinkTop V50, V70



Version 1.0 1/2025

Contents

1. Preliminary notes	2
2. Intended Use.....	3
3. Prerequisites.....	3
4. Integrate Two-Acyclic_nP or Four_AcyclicCmds_nP to the PLC	4
5. Acyclic Command Parameter Data Types and Descriptions	13
6. Requested Packet Interval (RPI) of the ifm IO-Link Ethernet module.....	15
7. Example to write and read a parameter of an IO-Link Device.....	16
8. Skipping Commands.....	23



1 Preliminary notes

Disclaimer of Warranties

Alfa Laval disclaims to the fullest extent authorized by law any and all warranties, whether express or implied, including, without limitation, any implied warranties of title, non-infringement, quiet enjoyment, integration, merchantability or fitness for a particular purpose.

Without limitation of the foregoing, Alfa Laval expressly does not warrant that:

1. The software will meet your requirements or expectations.
2. The software or the software content will be free of bugs, errors, viruses or other defects.
3. Any results, output, or data provided through or generated by the software will be accurate, up-to-date, complete or reliable.
4. The software will be compatible with third party software.
5. Any errors in the software will be corrected.

DEMO SOFTWARE AND TEMPLATES

Demo software and templates are provided “as is” and “as available”, without any warranty of any kind, either express or implied. The User acknowledges and agrees to use the software at User’s own risk. In no event shall Alfa Laval be held liable for any direct, indirect, incidental or consequential damages arising out of the use of or inability to use the software. User may use the software solely for demonstration purposes and to assess the software functionalities and capabilities.

Software written specifically for Customer

1. The software created and used is put together by Alfa Laval specially to the customer using modular software components made by Alfa Laval for numerous applications (Standard software modules) and adapted to the contractual service required (customer-specific application program).
2. Upon complete payment of the purchase price for the customer-specific application program, Alfa Laval transfers the non-exclusive, locally and temporarily unrestricted usage right thereof to the customer, without the customer acquiring any rights of any kind to the standard software module on which the individual or customer-specific adaptation is based. Notwithstanding these provisions, Alfa Laval reserves the right to produce and offer customer specific software solutions of the same kind for other customers based on other terms of reference. In any case Alfa Laval retains a simple right of usage of the customer-specific solution for internal purposes.
3. Upon acceptance of the program, User acknowledges and agrees to use the software at User’s own risk. User also acknowledges and agrees upon acceptance of the program that the software meets the requirements of the specifications agreed upon. Alfa Laval disclaims any and all warranties, in particular regarding fitness of the software for a particular purpose.



2 Intended Use

The Two_AcyclicCmds_nP.L5X or Four_AcyclicCmds_nP.L5X Add-On Instructions provide a logical user interface to read or write IO-Link ISDU commands. Each command can be set to read data from or to write data to an IO-Link device.

The Add-On Instruction is to be integrated in a PLC program to:

1. Sequentially send up to two or four ISDU commands.
2. Read data from an IO-Link device.
3. Write data to an IO-Link device.

3 Prerequisites

The user has integrated an ifm IO-Link master AL112x, AL122x, AL132x, AL142x, AL192x.ALRS Logix 5000 or Studio 5000 software.

4 Integrate Two_AcyclicCmds_nP or Four_AcyclicCmds_n to the PLC project's logic

Section 4.1 and 4.2 show the integration steps while offline with the PLC. It is also possible to perform the integration methods while online.

4.1 Import the Add-On Instruction

1. In the Controller Organizer, expand the **Assets** folder. Right click Add-On Instruction and select **Import Add-On Instruction...**

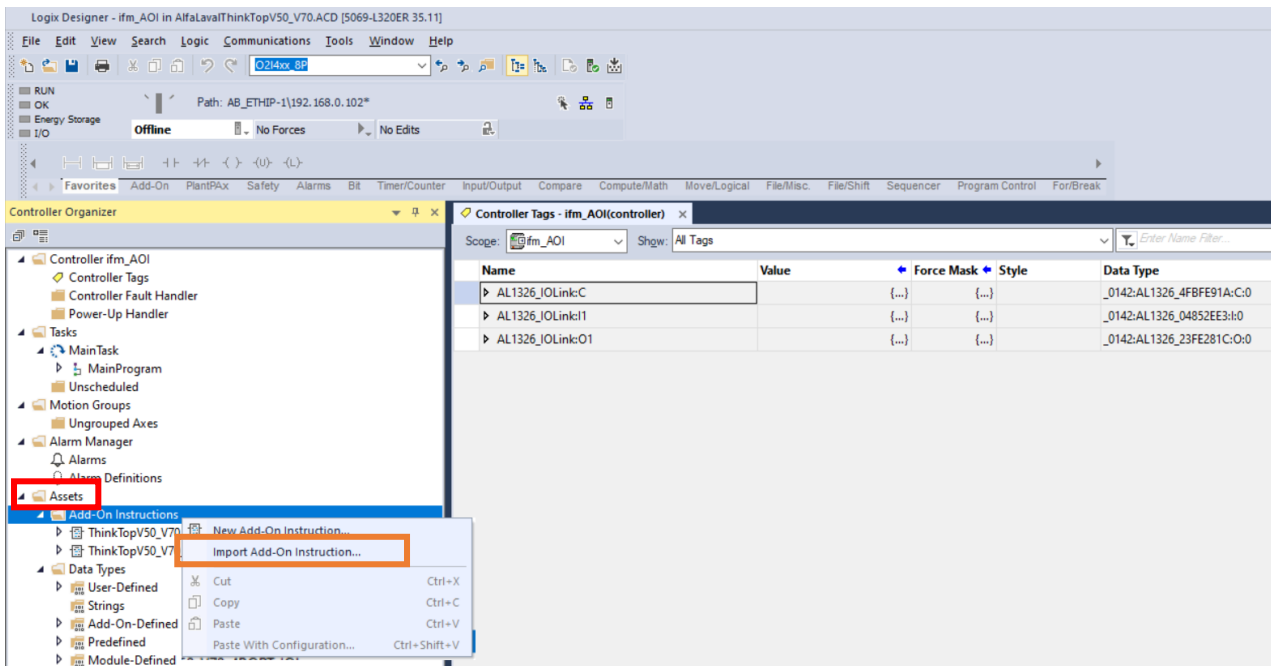


Figure 1

2. Navigate to the folder named **2. Add-On Instructions** in the ThinkTop_EIP_AllenBradley_StartupPackage. Open the folder to find a folder named **Read Write Acyclic Data**.

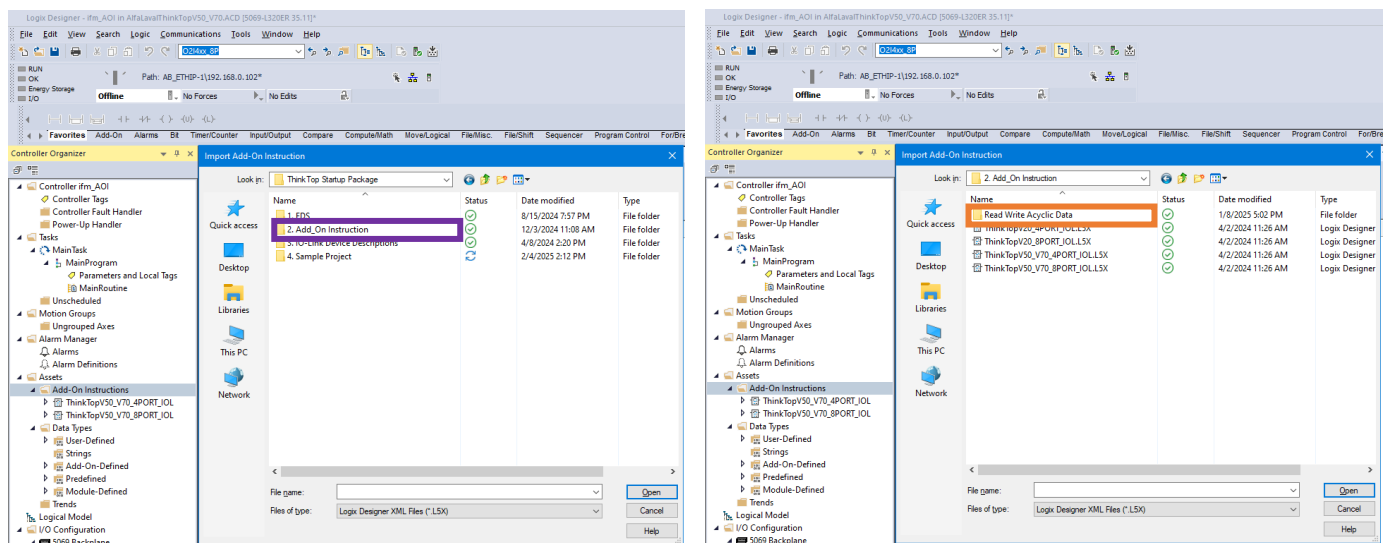


Figure 2

3. Select a Two or Four Acyclic Command Add-On Instruction and click **Open**.

Two_AcyclicCmds_nP.L5X processes up to two commands, command 1 and command 2, in order.

Four_AcyclicCmds_nP.L5X processes up to four commands. Command 1..Command 4, in order.

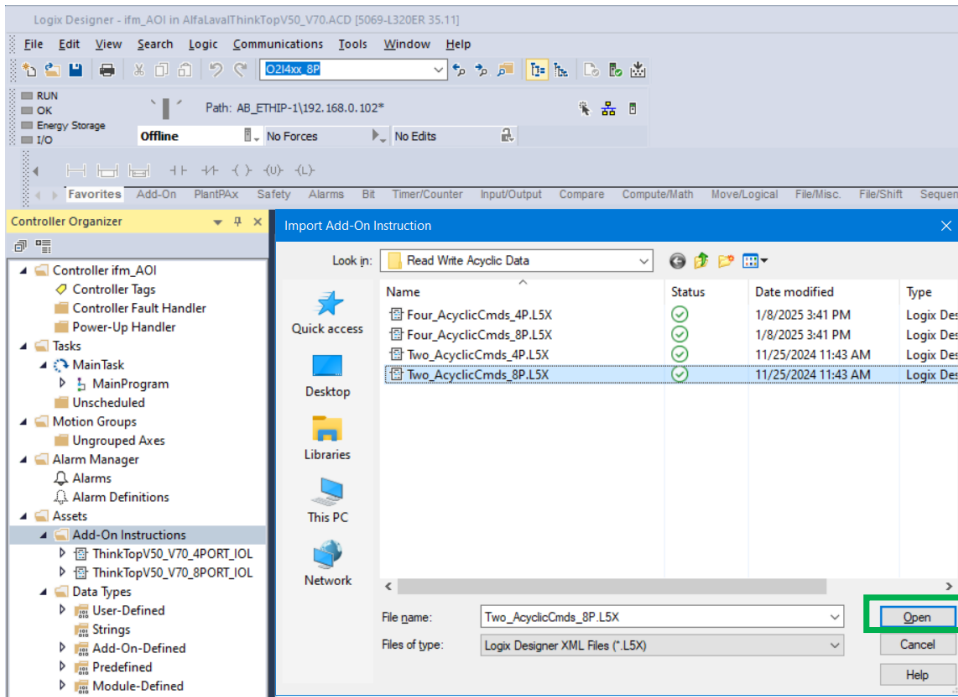


Figure 3

4. Click **OK** to Import.

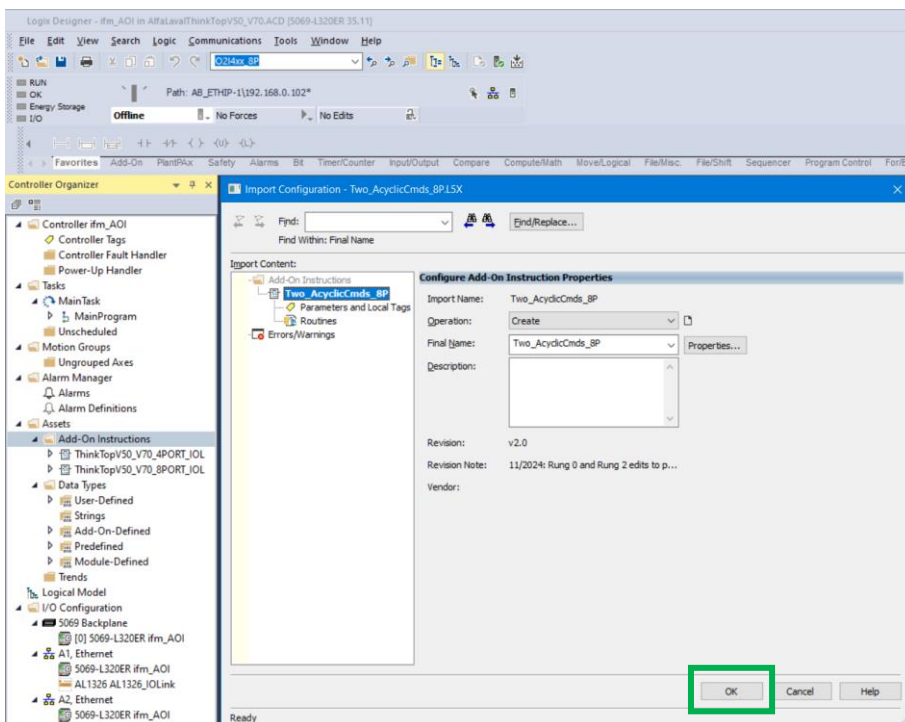
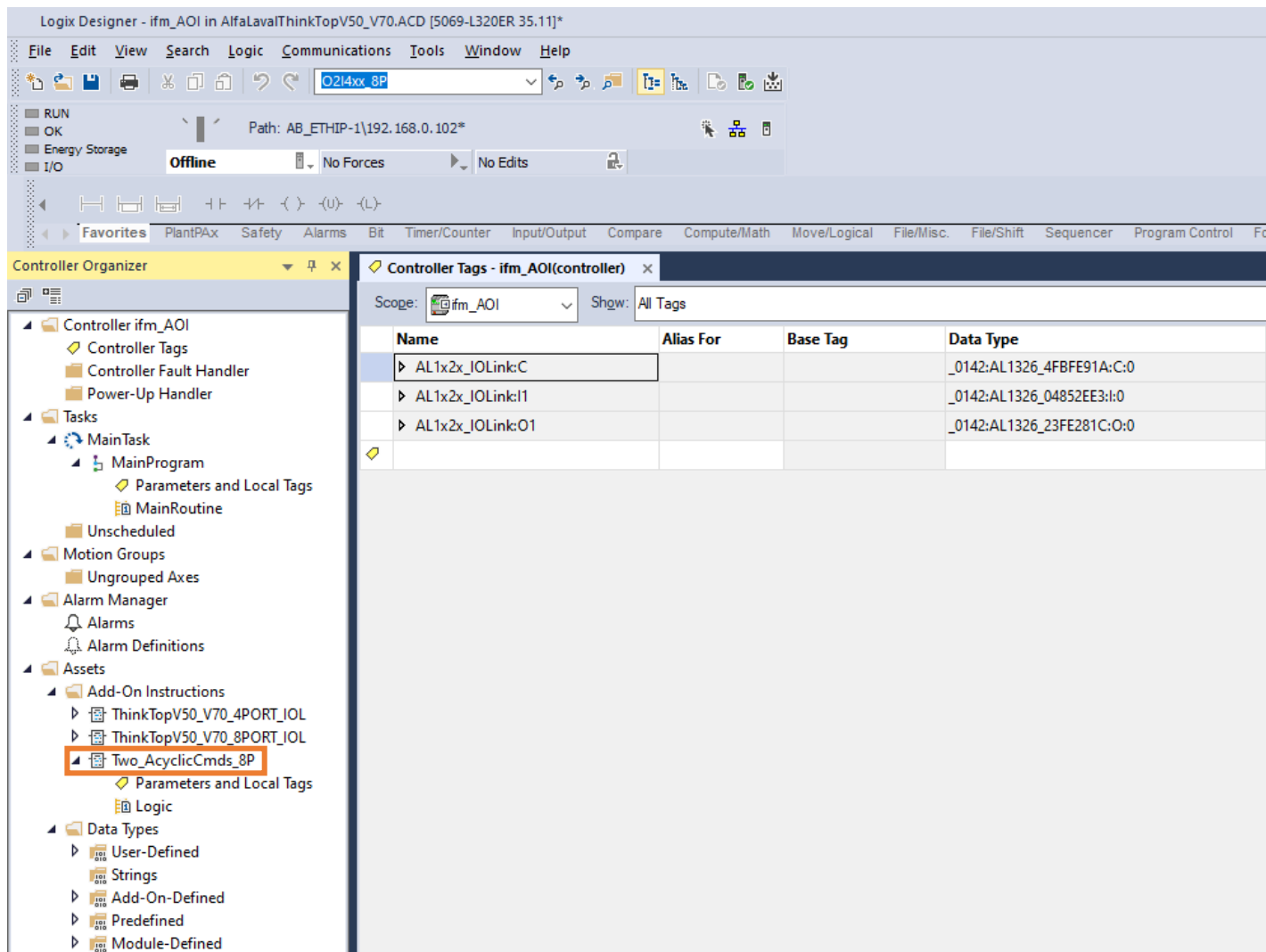


Figure 4

5. The **imported Add-On Instruction** is displayed in the Add-On Instructions folder.



The screenshot shows the Logix Designer interface. The **Controller Organizer** on the left lists the project structure. Under **Assets**, the **Add-On Instructions** folder is expanded, and **Two_AcyclicCmds_8P** is highlighted with a red box. The **Controller Tags - ifm_AOI(controller)** window on the right shows a table of tags.

Name	Alias For	Base Tag	Data Type
AL1x2x_IOLink:C			_0142:AL1326_4FBFE91A:C:0
AL1x2x_IOLink:I1			_0142:AL1326_04852EE3:I:0
AL1x2x_IOLink:O1			_0142:AL1326_23FE281C:O:0

Figure 5

4.2 Integrate the Acyclic Add-On Instruction in a routine

1. Click and drag the **Two_AcyclicCmd_nP** or **Four_AcyclicCmds_nP** Add-On Instruction onto a rung of logic.

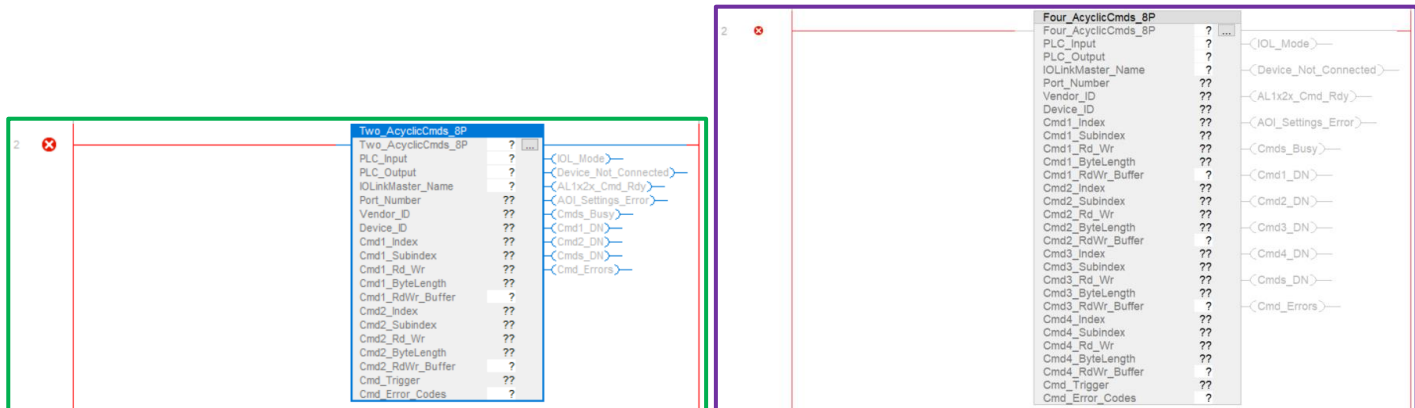


Figure 6

Note: The Two_Acyclic_8P Add-On Instruction will be used in this document. The integration and behavior of the IO-Link master, Two_Acyclic 4-port version and the Four_Acyclic Command Add-On Instructions is the same.

2. In the Two_AcyclicCmds_8P field, right click ? and select **New Tag**.

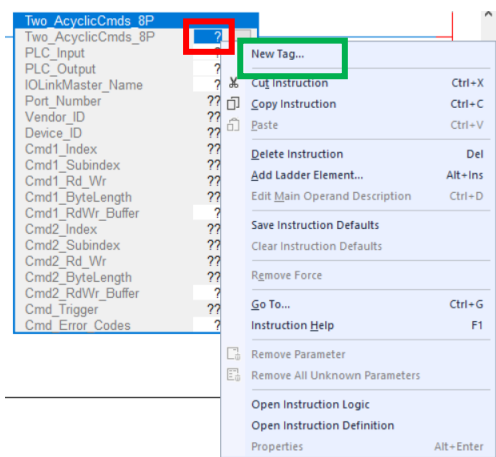


Figure 7

3. Create a **user defined name** and click **Create**.

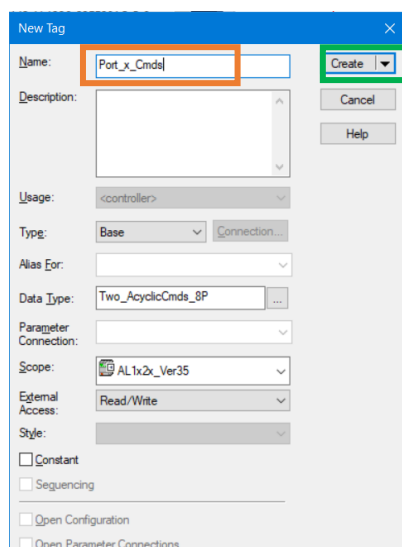


Figure 8

- Assign the PLC_Input and PLC_Output tags using the ifm IO-Link master's producing **input data array** and consuming **output data array** from the Controller Tags.

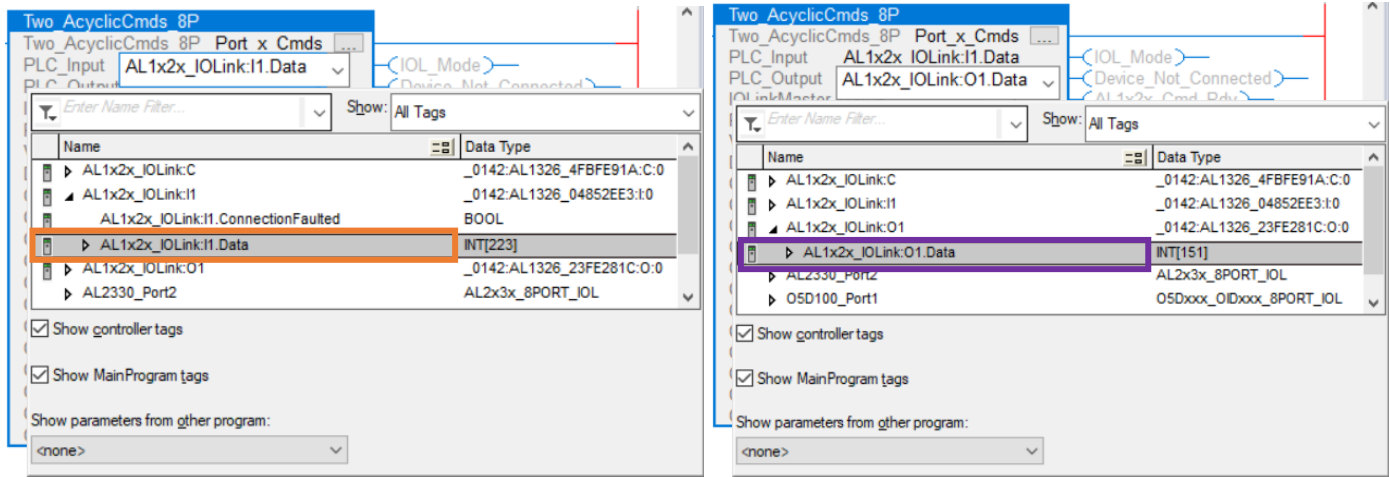


Figure 9

- In the IO-Link Master_Name field, right click ? and click **New Tag**.

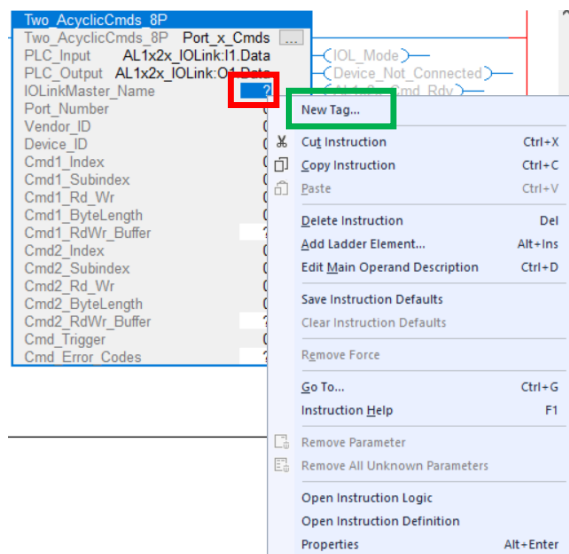
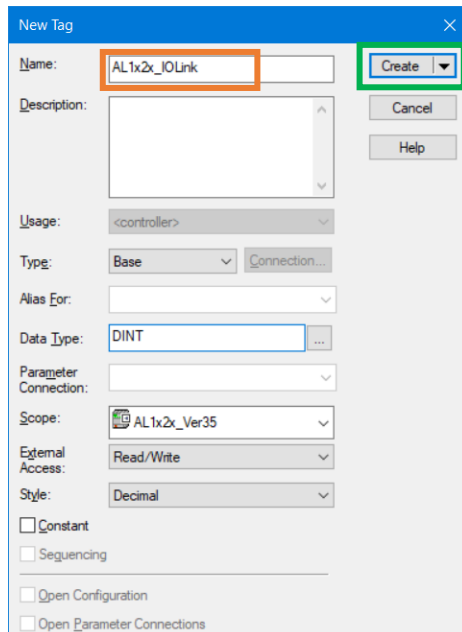


Figure 10



6. Create a **user defined name** and click **Create**.

Note: If multiple instances of the AcyclicCmds Add-On Instruction will be for other IO-Link devices on the same IO-Link master, the IOLinkMaster_Name tag will be used in each instance. It is suggested, but not required, to use the name given to the IO-Link master in the Ethernet/IP module setup for the name of this tag.



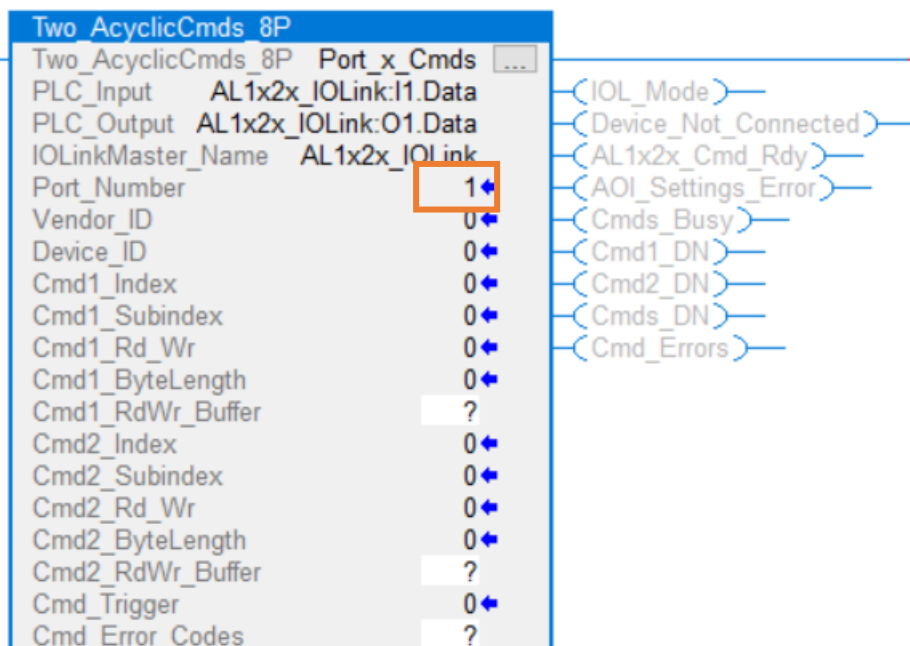
The 'New Tag' dialog box is shown with the following fields and values:

- Name: AL1x2x_IOLink
- Description: (empty)
- Usage: <controller>
- Type: Base
- Alias For: (empty)
- Data Type: DINT
- Parameter Connection: (empty)
- Scope: AL1x2x_Ver35
- External Access: Read/Write
- Style: Decimal
- Constant: ☐
- Sequencing: ☐
- Open Configuration: ☐
- Open Parameter Connections: ☐

The 'Create' button is highlighted with a green box.

Figure 11

7. Enter the port number of the IO-Link master where the IO-Link device is connected.
Value range=1..4 (four port masters) or **Value range =1..8** (eight port masters)



The 'Two_AcyclicCmds_8P' configuration window is shown with the following fields and values:

Field	Value
PLC_Input	AL1x2x_IOLink:I1.Data
PLC_Output	AL1x2x_IOLink:O1.Data
IOLinkMaster_Name	AL1x2x_IOLink
Port_Number	1
Vendor_ID	0
Device_ID	0
Cmd1_Index	0
Cmd1_Subindex	0
Cmd1_Rd_Wr	0
Cmd1_ByteLength	0
Cmd1_RdWr_Buffer	?
Cmd2_Index	0
Cmd2_Subindex	0
Cmd2_Rd_Wr	0
Cmd2_ByteLength	0
Cmd2_RdWr_Buffer	?
Cmd_Trigger	0
Cmd_Error_Codes	?

The 'Port_Number' field is highlighted with a red box. To the right of the configuration table, there are several output fields with blue lines connecting them to the configuration table:

- IOL_Mode
- Device_Not_Connected
- AL1x2x_Cmd_Rdy
- AOI_Settings_Error
- Cmds_Busy
- Cmd1_DN
- Cmd2_DN
- Cmds_DN
- Cmd_Errors

Figure 12

8. For each Cmd*_RdWr_Buffer (*=1..4), right click ? and select **New Tag**.

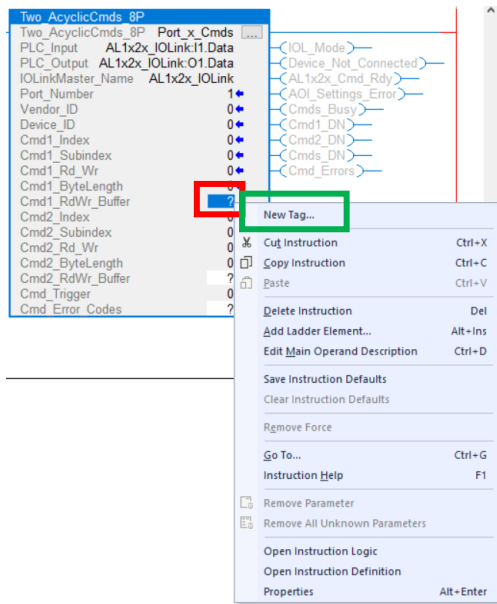


Figure 13

9. Create a unique **user defined name** for each Cmd*_RdWr_Buffer and click **Create**.

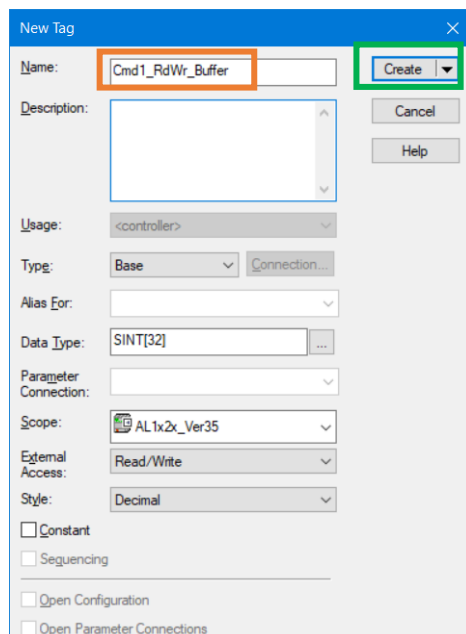


Figure 14

10. In the Cmd_Errors_Codes field, right click ? and select **New Tag**.

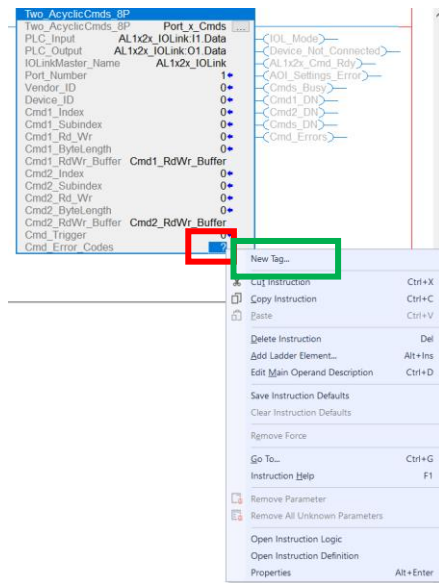


Figure 15

11. Create a **user defined name** and click **Create**.

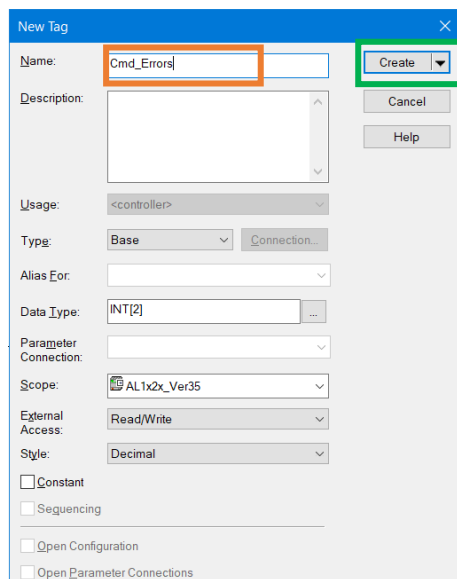


Figure 16

12. The rung is complete with no errors.

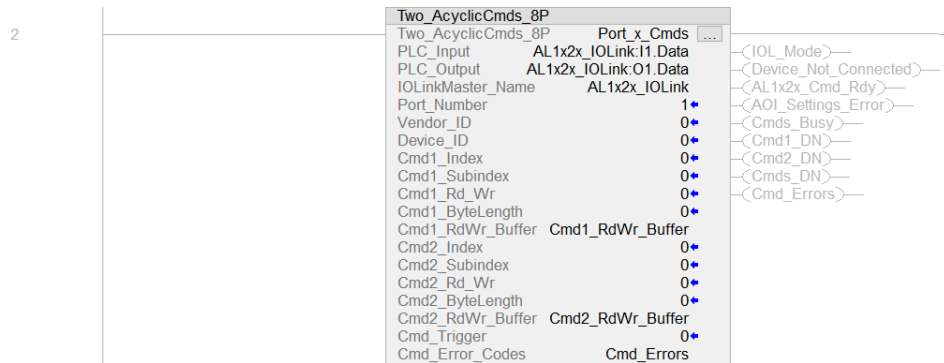


Figure 17

13. Enter the command information as needed. See section 5 and 7 for an explanation of the user inputs.

14. If the previous steps were made offline, the Add-On Instruction is ready to be downloaded to the PLC.
If the previous steps were made online, the pending edits are ready to be finalized.

5. Acyclic Command Parameter Data Types and Descriptions

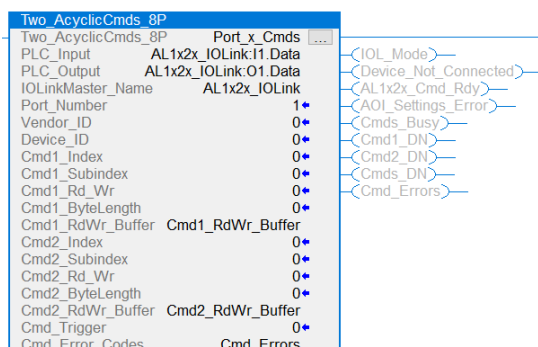


Figure 18

5.1 Add-On Instruction Input Parameter tags: Values consumed by the PLC

Note: An * represents the command number.

IOLinkMasterName: User created tag of type DINT.

The tag internally controls the AL1x2x_Cmd_Rdy output bit and references other instances of the acyclic Add-On Instruction used for other devices on the same IO-Link master.

If multiple instances of the AcyclicCmds Add-On Instruction are to be integrated into the project, use the same IOLinkMaster_Name in each instance.

It is suggested, but not required, to use the name given to the IO-Link master in the Ethernet/IP module setup for the name of this tag.

Port_Number: User input of the port number (SINT) where the IO-Link device is connected on the IO-Link master.

Cmd*_Index: User input of command Index (INT). See the IO-Link device's documentation for a listing of possible values.

A value =0 will ignore Cmd* processing. See section 8 Skipping Commands.

Cmd*_SubIndex: User input of command Subindex (INT). See the IO-Link device's documentation for a listing of possible values for the Index being used.

Cmd*_Rd_Wr : User input of command Index (SINT)

Value range – 1..2 | Default =1

Set =1 to request a read command

Set =2 to request a write command

Cmd*_ByteLength: User input of the byte length of the command to be written (SINT). See the IO-Link Device's documentation for a listing of possible values for the Index being used.

Value range – 1..32

Cmd_Trigger: User controlled binary setting (BOOL). A rising edge starts the command processing.

5.2 Add-On Instruction Output Parameter tags: Values produced to the PLC

Vendor_ID: Static integer (INT) value produced by the IO-Link device while connected to the IO-Link master

For Alfa Laval products value =1292 decimal / 0x050C hex

Device_ID: Static double integer (DINT) value produced while IO-Link device is connected to the IO-Link master

Note: The vendor and device ID values are documented in the IO-Link Device's documentation.

IOL_Mode: Binary (BOOL) state of the operating mode of the IO-Link master port.

Value range – TRUE (IO-Link mode) .. FALSE (port has been set to disabled, digital input or digital output mode)

Note: The eds file installation defaults all ports to IO-Link mode. The port mode may be changed in the configuration controller tags.

Device_Not_Connected: Binary (BOOL) state of the IO-Link's devices connection status of the Port_Number.

Value range – FALSE (an IO-Link device is connected) .. TRUE (an IO-Link device is not connected)



AOI_Settings_Error: Binary (BOOL) state of at least one, Add-On Instruction user input error.
Value Range – FALSE (No error present) .. TRUE (at least one error present)

The following cases will set the AOI_Settings_Error =TRUE.

1. Port_Number ≤0 or >4 if using Two_AcyclicCmds_4P or Four_AcyclicCmds_4P
Port_Number ≤0 or >8 if using Two_AcyclicCmds_8P or Four_AcyclicCmds_8P
2. Cmd*_Index ≠0 and Cmd*_Rd_Wr ≠1 or 2
4. Cmd*_Index ≠0 and Cmd*_Rd_Wr =2 and Cmd*_ByteLength ≤0
5. Cmd*_ByteLength >32

If AOI_Settings_Error =TRUE then the AL1x2x_CmdRdy bit is set FALSE and the Cmd_Trigger has no function.
The AOI_Setting_Error output =FALSE when no user input errors are present.

AL1x2x_Cmd_Rdy: Binary (BOOL) state of the IO-Link master command mailbox.
Value range – TRUE: The IO-Link master is ready for a user input of Cmd_Trigger.
FALSE: The IO-Link master's mailbox is in use. The Cmd_Trigger has no function.

Cmds_Busy: Binary (BOOL) indication of commands processing.
Value range – TRUE: Commands are processing.
FALSE: Commands are not processing.

Cmd*_DN: Binary (BOOL) state of the completion of each command number.
Value range – TRUE: The command number has completed processing.
Value range – FALSE: The command number has not completed processing, the command number was skipped or the command number has resulted in an error.

Cmds_DN: Binary (BOOL) state of the completion of all command numbers.
Value range – TRUE: All command numbers have completed processing.
Value range – FALSE: All command numbers have not completed processing.

Cmd_Errors: Binary (BOOL) state of the presence of one or more command errors.
Value range – TRUE: At least one command error has resulted from the command processing.
FALSE: No command errors have resulted from the command processing.

Note: When the IO-Link device is disconnected from the IO-Link master port Cmds_Busy, Cmd*_DN, Cmds_DN, Cmd_Errors is set FALSE.

5.3 Add-On Instruction InOutParameter tags: Values produced or consumed by the PLC

PLC_Input: Reference to the Input Controller tag <IOLinkMasterName>:I1.Data

PLC_Output: Reference to the Output Controller tag <IOLinkMasterName>:O1.Data

Cmd*_RdWrBuffer: User created tag array of type SINT of size 32.

If Cmd*_RdWr =1 (request read command), then the Cmd*_RdWrBuffer is used to receive the response data from the IO-Link device.

If Cmd*_RdWr =2 (request write command), then the data values to be written to the IO-Link device are set in the Cmd*_RdWrBuffer. See the IO-Link Device's documentation for a listing of possible values for the Index being used.

Cmd_Errors_Codes: User created tag array of INT of size 2 or 4 (depending on two or four command Add-On instruction being used).

Cmd_Errors_Codes[0] = Error code buffer for command 1

Cmd_Errors_Codes[1] = Error code buffer for command 2

Cmd_Errors_Codes[2] = Error code buffer for command 3^A

Cmd_Errors_Codes[3] = Error code buffer for command 4^A

^A Only used with Four_AcyclicCmd_nP Add-On Instructions

The command error integers are refreshed on the rising edges of the Cmd_DN output bit.
See Appendix A for a listing of command errors.

6 Requested Packet Interval (RPI) of the ifm IO-Link Ethernet module

The RPI time of the IO-Link master Ethernet/IP module is the polling rate in which the PLC will update the input and output controller tags. The default setting is 10 ms. When using the acyclic command Add-On Instruction the setting range may be set from 1..40ms.

1. To adjust the RPI time, right click on the **Ethernet/IP module** and select **Properties**.

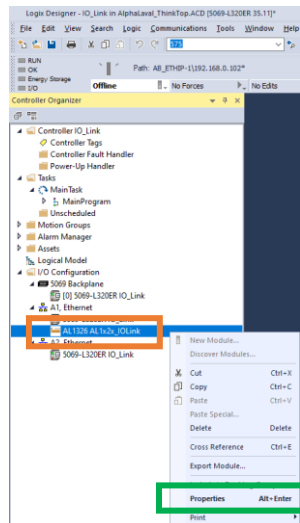


Figure 19

2. In the **Module Properties** window, select **Connection** and adjust the **RPI** field within the 1.0..40.0 ms range. Click **OK** to accept the change.

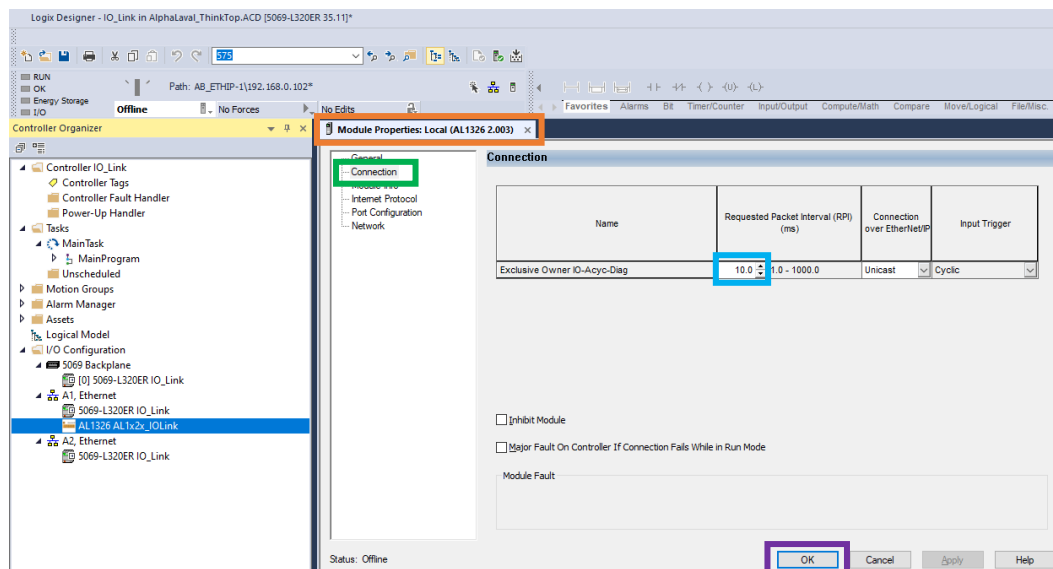


Figure 20

7 Example to write and read a parameter of an IO-Link device

Example: A ThinkTop V70 is connected to port 1 of the AL1x2x.

Use command one to send a command to set the travel distance of the seat during a pulse from De-En and Main-En position.

Use command two to read the travel distance of the seat during a pulse from De-En and Main-En position.



Figure 21

1. Navigate to the folder named **3. IO-Link Device Descriptions** in the ThinkTop_EIP_AllenBradley_StartupPackage. Open the folder to find the **IO-Link interface description PDF** file for the ThinkTop V70.

Name	Status	Date modified	Type	Name	Date modified	Type
1. EDS	✓	8/15/2024 7:57 PM	File folder	interface-description-thinktop-v20-io-link.pdf	4/8/2024 2:20 PM	Adobe Acroba...
2. Add-On-Instruction	✓	12/3/2024 11:08 AM	File folder	interface-description-thinktop-v50-io-link-after-dec.-2021-.pdf	1/10/2024 11:39 AM	Adobe Acroba...
3. IO-Link Device Descriptions	✓	4/8/2024 2:20 PM	File folder	interface-description-thinktop-v70-io-link-after-dec.-2021-.pdf	1/10/2024 4:21 PM	Adobe Acroba...
4. Sample Project	✓	2/6/2025 8:33 AM	File folder			
ChangeLog.txt	✓	10/30/2024 10:45 AM	Text Document			

Figure 22

2. The LED color customization, RGB-Colour, read/write command is found in section 6 of the PDF document. The information needed to be input in the TwoAcyclicCmds Add-On Instruction is the **Index**, **Subindex**, **Command Length** and **Value range** for the various colors option.

RGB-Colour	Index 74	Subindex 0	RecordT (64 Bit)	ReadWrite
Option for customisation of the colours for the visual feedback				
Button press		Subindex 1	UIntegerT (8 Bit)	
RGB colour for button press feedback				
Factory setting	5	(Purple)		
Value range	0	(RGB off)		
	1	(Blue)		
	2	(Green)		
	3	(Cyan)		
	4	(Red)		
	5	(Purple)		
	6	(Yellow)		
	7	(White)		
Error		Subindex 2	UIntegerT (8 Bit)	
RGB colour for error feedback				
Factory setting	4	(Red)		
Value range	0	(RGB off)		
	1	(Blue)		
	2	(Green)		
	3	(Cyan)		
	4	(Red)		
	5	(Purple)		
	6	(Yellow)		
	7	(White)		
Out		Subindex 3	UIntegerT (8 Bit)	
RGB colour for Moving (out of any position)				
Factory setting	0	(RGB off)		
Value range	0	(RGB off)		
	1	(Blue)		
	2	(Green)		
	3	(Cyan)		
	4	(Red)		
	5	(Purple)		
	6	(Yellow)		
	7	(White)		
De-En		Subindex 4	UIntegerT (8 Bit)	
RGB colour for De-Energized position				
Factory setting	2	(Green)		
Value range	0	(RGB off)		
	1	(Blue)		
	2	(Green)		
	3	(Cyan)		
	4	(Red)		
	5	(Purple)		
	6	(Yellow)		
	7	(White)		
Main		Subindex 5	UIntegerT (8 Bit)	
RGB colour for Main position				
Factory setting	7	(White)		
Value range	0	(RGB off)		
	1	(Blue)		
	2	(Green)		
	3	(Cyan)		
	4	(Red)		
	5	(Purple)		
	6	(Yellow)		
	7	(White)		
Usl		Subindex 6	UIntegerT (8 Bit)	
RGB colour for Upper Seat Lift position				
Factory setting	3	(Cyan)		
Value range	0	(RGB off)		
	1	(Blue)		
	2	(Green)		
	3	(Cyan)		
	4	(Red)		
	5	(Purple)		
	6	(Yellow)		
	7	(White)		
Lsp		Subindex 7	UIntegerT (8 Bit)	
RGB colour for Lower Seat Push position				
Factory setting	6	(Yellow)		
Value range	0	(RGB off)		
	1	(Blue)		
	2	(Green)		
	3	(Cyan)		
	4	(Red)		
	5	(Purple)		
	6	(Yellow)		
	7	(White)		

Figure 23

3. Enter the **Port Number** where the IO-Link device to receive the command request is connected to the IO-Link master.

4. Enter the write request data for command one: **Index**, **Subindex**, **Write Command=2**, and **ByteLength**.
Note: In the example the Subindex =0 because all LED colors will be set with one command. It is also possible to set each LED color function individually. In this case the Subindex =1..7.

Note: In the example the ByteLength =8 (64 bits) because the LED colors will be set with one command. If using Subindexes for setting LED color functions individually then ByteLength =1 (8 bits).

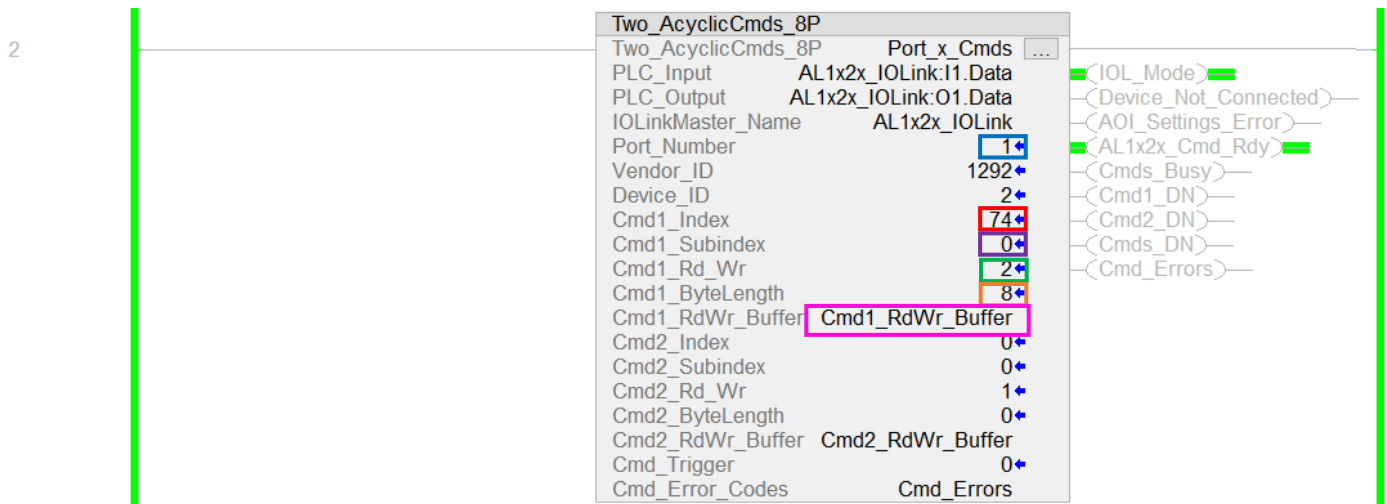


Figure 24

5. Right click on the **Cmd1_RdWr_Buffer** array tag and monitor the tag values.

Enter the **color values for each LED function**. Byte[0]=Button Press color...Byte[6]= Lsp color

Name	Value	Force Mask	Style	Data Type
▲ Cmd1_RdWr_Buffer	{...}	{...}	Decimal	SINT[32]
▶ Cmd1_RdWr_Buffer[0]	1		Decimal	SINT
▶ Cmd1_RdWr_Buffer[1]	2		Decimal	SINT
▶ Cmd1_RdWr_Buffer[2]	3		Decimal	SINT
▶ Cmd1_RdWr_Buffer[3]	4		Decimal	SINT
▶ Cmd1_RdWr_Buffer[4]	5		Decimal	SINT
▶ Cmd1_RdWr_Buffer[5]	6		Decimal	SINT
▶ Cmd1_RdWr_Buffer[6]	7		Decimal	SINT

Figure 25

6. Enter the read request data for command two: **Index**, **Subindex**, **Read Command=1**, **ByteLength=0**.
 ByteLength=0, no length is sent to the IO-Link device for read commands.
 No data is set to the **Cmd2_ByteLength_Buffer** for read command. It is used for receiving the data read from the IO-Link device when the command has finished processing.

2

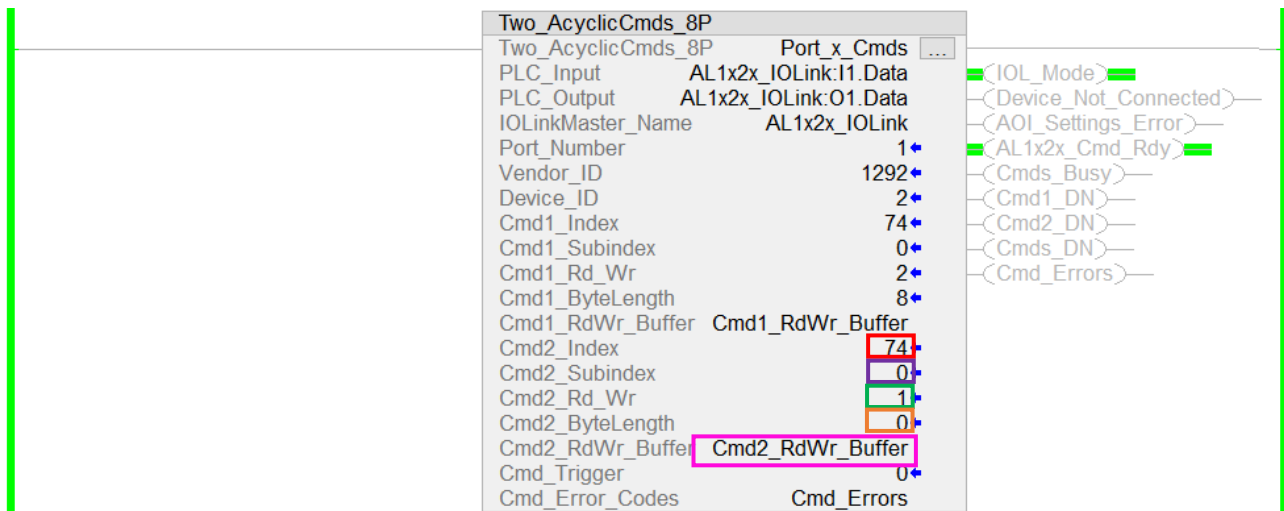


Figure 26

7. Set the rising edge of the **Cmd_Trigger** bit to start the commands.
 Note: Cmd_Trigger starts the commands via a rising edge, i.e., used with a ONS instruction.
 The Cmd_Trigger can also be toggled true as shown below. See point d in section 7.1 for a description of the behavior of the Add-On Instruction outputs bits.

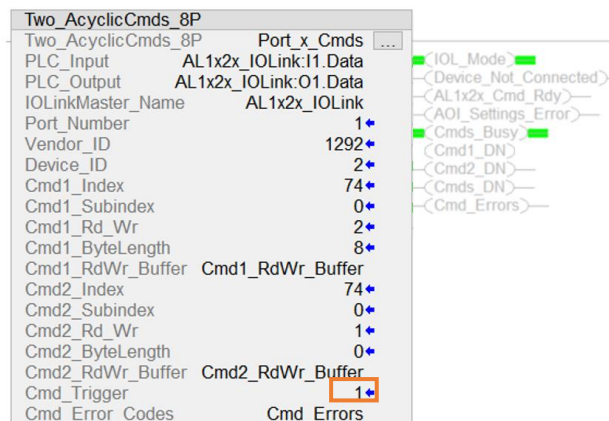


Figure 27

7.1 Processing behavior

1. Processing behavior

a. On the rising edge of Cmd_Trigger =TRUE

Cmds_Busy =TRUE
 AL1x2x_Cmd_Rdy =FALSE
 Cmd1_DN =FALSE
 Cmd2_DN =FALSE
 Cmds_DN =FALSE
 Cmd_Errors =FALSE

b. When Cmd1 is done without error

Cmds_Busy =TRUE
 AL1x2x_Cmd_Rdy =FALSE
 Cmd1_DN =TRUE
 Cmd2_DN =FALSE
 Cmds_DN =FALSE
 Cmd_Errors =FALSE

When Cmd1 is done with error

Cmds_Busy =TRUE
 AL1x2x_Cmd_Rdy =FALSE
 Cmd1_DN =FALSE
 Cmd2_DN =FALSE
 Cmds_DN =FALSE
 Cmd_Errors =TRUE

c. When Cmd2 is done without error

Cmds_Busy =FALSE
 AL1x2x_Cmd_Rdy =FALSE
 Cmd1_DN =TRUE/FALSE
 Cmd2_DN =TRUE
 Cmds_DN =TRUE
 Cmd_Errors =TRUE/FALSE

When Cmd2 is done with error

Cmds_Busy =FALSE
 AL1x2x_Cmd_Rdy =FALSE
 Cmd1_DN =TRUE/FALSE
 Cmd2_DN =FALSE
 Cmds_DN =TRUE
 Cmd_Errors =TRUE

d. Cmds_DN =TRUE and Cmd_Trigger =FALSE

Cmds_Busy =FALSE
 AL1x2x_Cmd_Rdy =TRUE
 Cmd1_DN =TRUE/FALSE
 Cmd2_DN =TRUE/FALSE
 Cmds_DN =TRUE
 Cmd_Errors =TRUE/FALSE

Cmds_DN =TRUE and Cmd_Trigger =TRUE

Cmds_Busy =FALSE
 AL1x2x_Cmd_Rdy =FALSE
 Cmd1_DN =TRUE/FALSE
 Cmd2_DN =TRUE/FALSE
 Cmds_DN =TRUE
 Cmd_Errors =TRUE/FALSE

Commands have process with no errors.

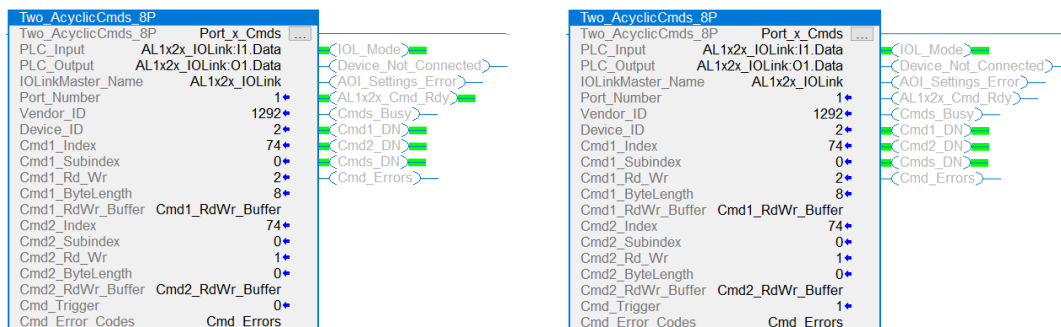


Figure 28

7.2 Command Errors Codes

Commands have processed and Cmd1 resulted in an **error**.

1. Right click on the **Cmd_Errors** array tag and monitor the tag values.

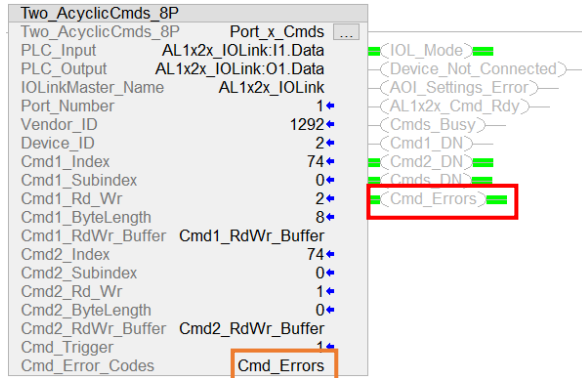


Figure 29

2. Cmd_Errors_Codes[0] has a **value** $\neq 0$, indicating command one resulted in an error.
3. Change the **Style** of the Cmd_Errors array to Hexidecmial.

Name	Value	Force Mas	Style	Data Type	Description
Cmd_Errors		[...]	Hex	INT[2]	
Cmd_Errors[0]	16#8030		Hex	INT	
Cmd_Errors[1]	16#0000		Hex	INT	

Figure 30

Cmd_Errors_Codes[0] = Error code buffer for command 1
 Cmd_Errors_Codes[1] = Error code buffer for command 2
 Cmd_Errors_Codes[2] = Error code buffer for command 3^A
 Cmd_Errors_Codes[3] = Error code buffer for command 4^A

^A Only used with Four_AcyclicCmd_nP Add-On Instructions

4. Reference the error code in section 9 of the IO Device Description PDF document.

9 Error types

Code	Name	Description
0x8000 32768d	Device application error - no details	Service was denied by the technology-specific application. No detailed root-cause information is available.
0x8011 32785d	Index not available	Read or write access attempt to a non-existing index.
0x8012 32786d	Subindex not available	Read or write access attempt to a non-existing subindex of an existing index.
0x8020 32800d	Service temporarily not available	Parameter not accessible due to the current state of the technology-specific application.
0x8023 32803d	Access denied	Write access to a read-only parameter or read access to write-only parameter.
0x8030 32816d	Parameter value out of range	Written parameter value is outside of the permitted value range.
0x8033 32819d	Parameter length overrun	Written parameter is longer than specified.
0x8034 32820d	Parameter length underrun	Written parameter is shorter than specified.
0x8035 32821d	Function unavailable	Written command is not supported by the technology-specific application
0x8036 32822d	Function temporarily unavailable	Written command is unavailable due to the current state of the technology-specific application.
0x8040 32832d	Invalid parameter set	Written single parameter value collides with other existing parameter settings.
0x8041 32833d	Inconsistent parameter set	Parameter set inconsistencies at the end of block parameter transfer. Device plausibility check failed.
0x8082 32898d	Application not ready	Read or write access denied. The technology-specific application is temporarily unavailable.



Error types are used for the ISDU response. Values unequal '0' indicate the cause of a failed ISDU read or write service.

Figure 31



5. The error 0x8030 indicates the written parameter value was out of range. The possible values for the LED color functions are 0..7 as shown in figure 23. The error was generated because a **value of 8** was processed by the command one.

Cmd1_RdWr_Buffer	{...}	{...}	Decimal	SINT[32]
▶ Cmd1_RdWr_Buffer[0]	8		Decimal	SINT
▶ Cmd1_RdWr_Buffer[1]	2		Decimal	SINT
▶ Cmd1_RdWr_Buffer[2]	3		Decimal	SINT
▶ Cmd1_RdWr_Buffer[3]	4		Decimal	SINT
▶ Cmd1_RdWr_Buffer[4]	5		Decimal	SINT
▶ Cmd1_RdWr_Buffer[5]	6		Decimal	SINT
▶ Cmd1_RdWr_Buffer[6]	7		Decimal	SINT

Figure 32

8 Skipping Commands

The rising edge of the Cmd_Trigger input bit starts the command processing starting at command one. It is possible to skip a command by entering a value =0 in the Cmd*_Index. Any command without an command Index =0 will be processed.

When a command Index =0, all other settings in the command may remain in their fields.

Example:

The Two_AcyclicCmds_8P Add-On Instruction is used with two write commands.

Command one: Set the color function using the RGB-Colour command.

Command two: Read the Log of the ThinkTop V70.

1. The command data is set for Index 74 in the Cmd1_RdWr_Buffer[0]=0 for each LED color function.

Name	Value	Force Mask	Style	Data Type
Cmd1_RdWr_Buffer	{...}	{...}	Decimal	SINT[32]
Cmd1_RdWr_Buffer[0]	1		Decimal	SINT
Cmd1_RdWr_Buffer[1]	2		Decimal	SINT
Cmd1_RdWr_Buffer[2]	3		Decimal	SINT
Cmd1_RdWr_Buffer[3]	4		Decimal	SINT
Cmd1_RdWr_Buffer[4]	5		Decimal	SINT
Cmd1_RdWr_Buffer[5]	6		Decimal	SINT
Cmd1_RdWr_Buffer[6]	7		Decimal	SINT

Figure 33

2. Cmd2_RdWr_Buffer[0] will be used to receive the Log command response using Index 80.

3. Set the Cmd1_Index =0 to prevent command one from processing.

4. The rising edge of the Cmd_Trigger starts the processing of the commands.

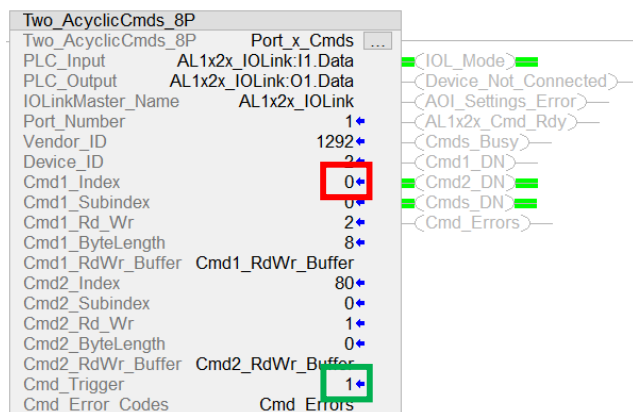


Figure 34

5. When Cmds_DN =TRUE & Cmd_Trigger =TRUE

Cmds_Busy =FALSE
 AL1x2x_Cmd_Rdy =FALSE
 Cmd1_DN =FALSE
 Cmd2_DN =TRUE
 Cmds_DN =TRUE

- When Cmds_DN =TRUE & Cmd_Trigger =FALSE

Cmds_Busy =FALSE
 AL1x2x_Cmd_Rdy =TRUE
 Cmd1_DN =FALSE
 Cmd2_DN =TRUE
 Cmds_DN =TRUE



6. Set the **Cmd1_Index** =74
7. Set **Cmd2_Index**=0
8. The rising edge of the **Cmd_Trigger** starts the processing of the commands.

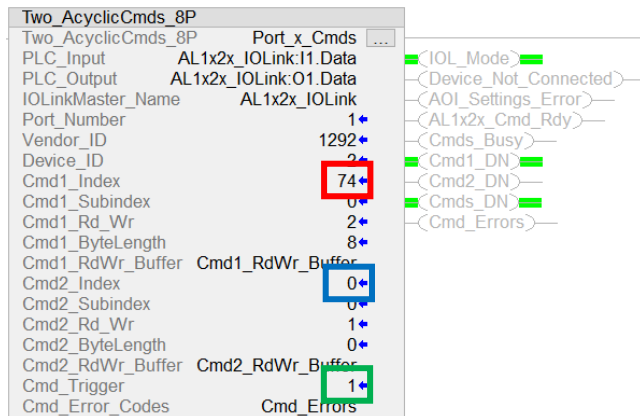


Figure 35

9. When Cmds_DN =TRUE & Cmd_Trigger =TRUE

Cmds_Busy	=FALSE
AL1x2x_Cmd_Rdy	=FALSE
Cmd1_DN	=TRUE
Cmd2_DN	=FALSE
Cmds_DN	=TRUE
- When Cmds_DN =TRUE & Cmd_Trigger =FALSE

Cmds_Busy	=FALSE
AL1x2x_Cmd_Rdy	=TRUE
Cmd1_DN	=TRUE
Cmd2_DN	=FALSE
Cmds_DN	=TRUE
10. Set **Cmd1_Index** =0 and **Cmd2_Index** =0 to prevent command one and two from processing but still allow the processing behavior of the user inputted Cmd_Trigger and Cmds_Busy and Cmds_DN output bits.
11. The rising edge of the **Cmd_Trigger** starts the processing of the commands.

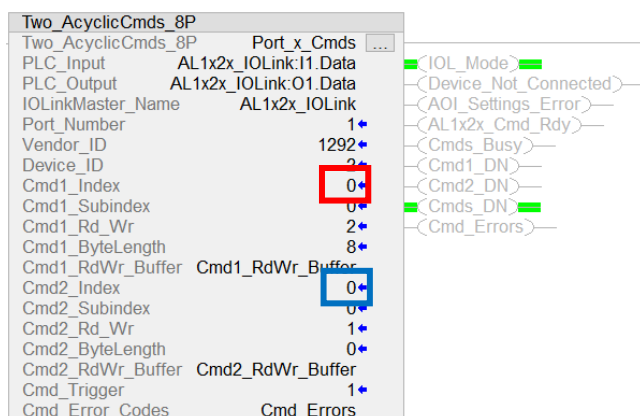


Figure 36

12. When Cmds_DN =TRUE & Cmd_Trigger =TRUE

Cmds_Busy	=FALSE
AL1x2x_Cmd_Rdy	=FALSE
Cmd1_DN	=FALSE
Cmd2_DN	=FALSE
Cmds_DN	=TRUE
- When Cmds_DN =TRUE & Cmd_Trigger =FALSE

Cmds_Busy	=FALSE
AL1x2x_Cmd_Rdy	=TRUE
Cmd1_DN	=FALSE
Cmd2_DN	=FALSE
Cmds_DN	=TRUE